ASP.NET MVC

CampusID
Sophia Antipolis

Jeremy Pekmez

Rodrigue Hajjar

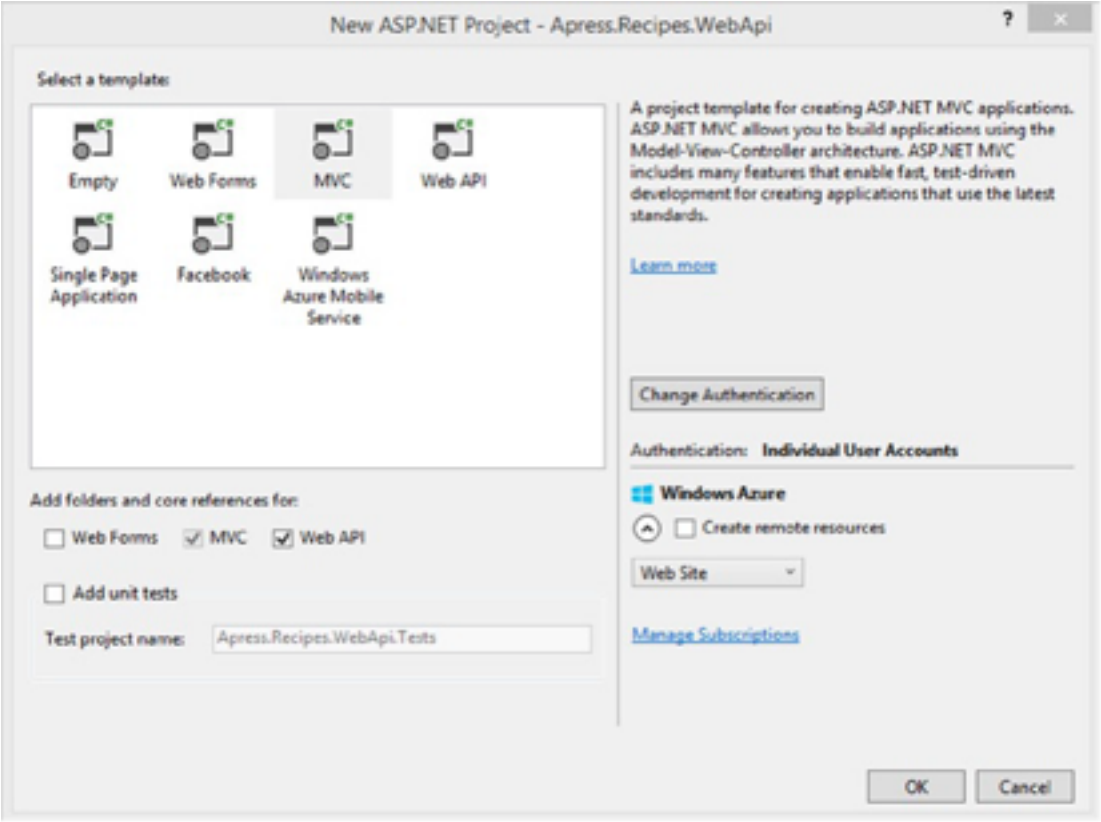
M1 / M2
2011-2012

Slides Roadmap

- ASP.NET MVC / API
- Self-Host ASP.NET Web API
- Scaffolding with ASP.NET Web API
- Specific Media Type from ASP.NET Web API
- Customize XML and JSON Responses
- OData
- Exceptions, Troubleshooting, and Documenting

ASP.NET MVC: API

- Integrate ASP.NET Web API into your ASP.NET MVC project



New ASP.NET Project - Apress.Recipes.WebApi

Select a template:

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Authentication: **Individual User Accounts**

Windows Azure

□ Create remote resources

Web Site

Add folders and core references for:

□ Web Forms ☑ MVC ☑ Web API

□ Add unit tests

Test project name: Apress.Recipes.WebApi.Tests

ASP.NET MVC: API

- Add Web API to any existing MVC project by installing it from NuGet.
  - Install-Package Microsoft.AspNet.WebApi

- Both approaches are equivalent

ASP.NET MVC: API

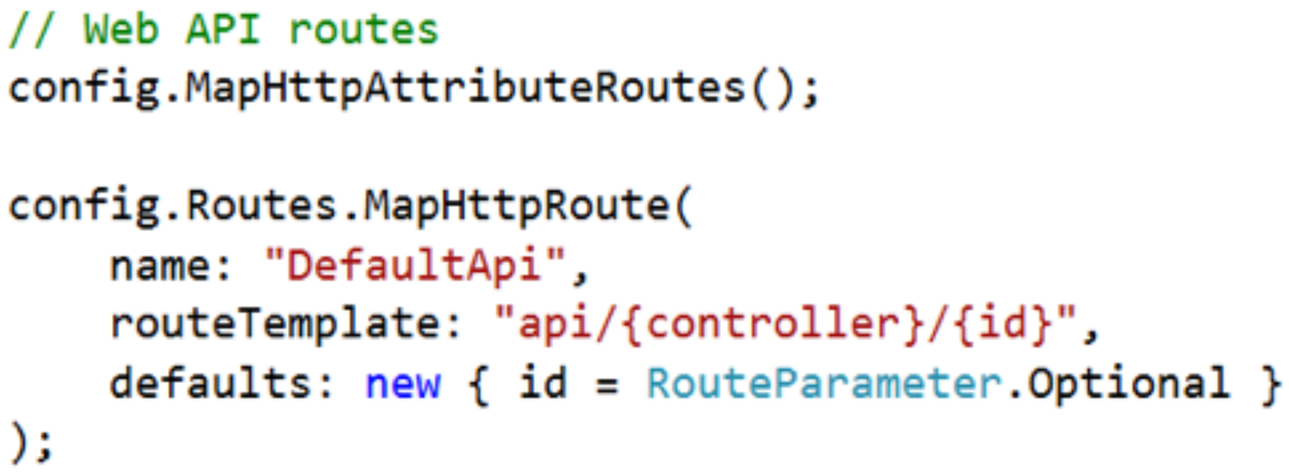
- ASP.NET Web API is build around System.Web.AsyncHandler

- Same architecture MVC / API

- Run in the same process
  - Share state as Global.asax events
  - Same web.config
  - Same Model

ASP.NET MVC: API

- ASP.NET Web API occupy URI /api



```
// Web API routes
config.MapHttpAttributeRoutes();

config.Routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

ASP.NET MVC: API

- Conventions:
  - Action Name {http verb}MethodName
  - Or attribute [HttpGet], [HttpPost]…
- Web Api with Entity Framework:
  - Code generation
  - CRUD operations
  - Views

ASP.NET MVC: API

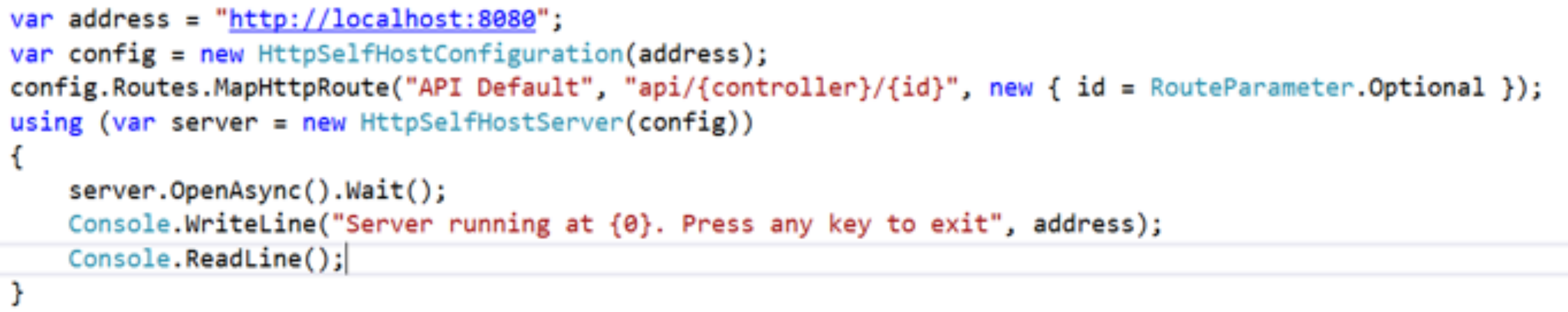
- Demo

Self-Host ASP.NET Web API

- Web API without IIS
- Based on WCF
- Web API self-hosting is available on NuGet:
  - Install-Package Microsoft.AspNet.WebApi.SelfHost
  - Install-Package Microsoft.AspNet.WebApi.OwinSelfHost

Self-Host ASP.NET Web API

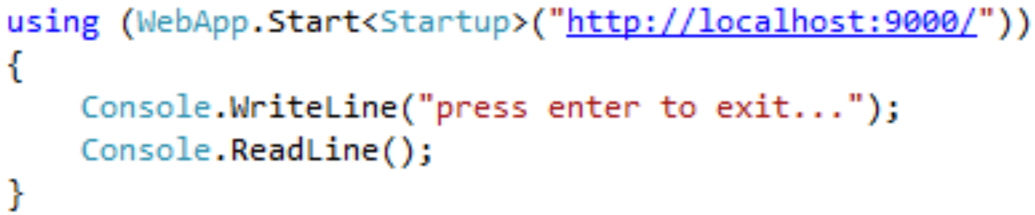
- Example configuration
  - using System.Web.Http.SelfHost
  - using System.Web.Http



```
var address = "http://localhost:8080";
var config = new HttpSelfHostConfiguration(address);
config.Routes.MapHttpRoute("API Default", "api/{controller}/{id}", new { id = RouteParameter.Optional });
using (var server = new HttpSelfHostServer(config))
{
    server.OpenAsync().Wait();
    Console.WriteLine("Server running at {0}. Press any key to exit", address);
    Console.ReadLine();
}
```

Self-Host ASP.NET Web API

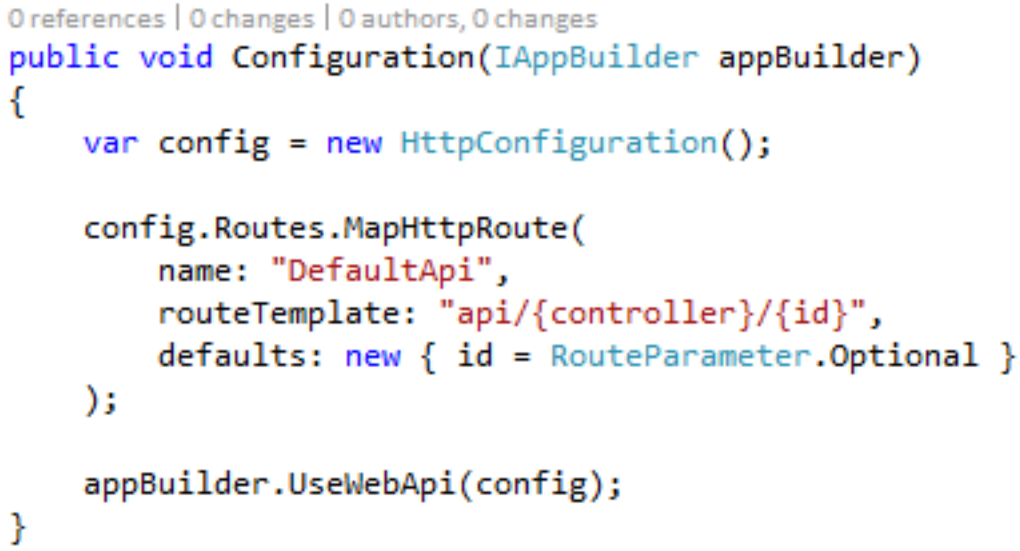
- Example configuration OWIN
  - class Program



```
using (WebApp.Start<Startup>("http://localhost:9000/"))
{
    Console.WriteLine("press enter to exit...");
    Console.ReadLine();
}
```

Self-Host ASP.NET Web API

- Example configuration OWIN
  - – class Startup



```
0 references | 0 changes | 0 authors, 0 changes
public void Configuration(IAppBuilder appBuilder)
{
    var config = new HttpConfiguration();

    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = RouteParameter.Optional }
    );

    appBuilder.UseWebApi(config);
}
```

Self-Host ASP.NET Web API

- Demo

Scaffolding with ASP.NET Web API

- T4-based code generation
- Template path (global effect):
  - C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\IDE\Extensions\Microsoft\Web\Mvc\Scaffolding\Templates
- Per projet: SideWaffle extensions

Scaffolding with ASP.NET Web API

- CRUD Operation:
  - GET /api/products => all products
  - GET /api/products/1 => product by id
  - GET /api/products?name=book => product by name
  - POST /api/products => add product
  - PUT /api/products/1 => update product
  - DELET /api/products/1 => delete product

Scaffolding with ASP.NET Web API

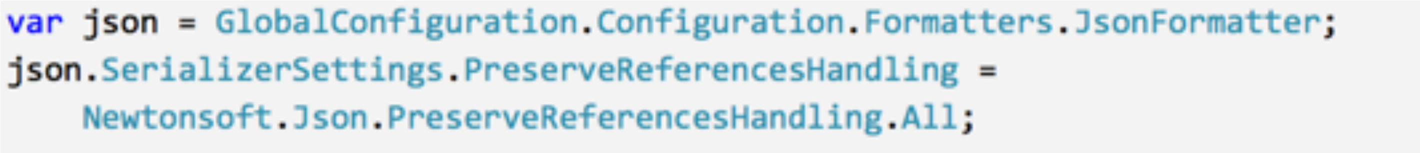
- Demo

ASP.NET MVC: API

- Default Formatter
  - GlobalConfiguration.Configuration.Formatters
  - JsonFormatter (Json.NET)
    - You can use Json.NET attribute like [JsonIgnore]…
    - Camel casing: CamelCasePropertyNamesContractResolver
  - XmlMediaTypeFormatter (DataContract serializer)
    - Yout can use DataContract Attribute like [DataMember]

ASP.NET MVC: API

- Circular references
  - Json



```
var json = GlobalConfiguration.Configuration.Formatters.JsonFormatter;
json.SerializerSettings.PreserveReferencesHandling =
    Newtonsoft.Json.PreserveReferencesHandling.All;
```

  - XML
  - [DataContract(IsReference=true)]

ASP.NET MVC: API

- Custom MediaTypeFormatter
  - MediaTypeFormatter (async read/write method)
  - BufferedMediaTypeFormatter (derives from MediaTypeFormatter and wrap read/write async method inside sync methods)

ASP.NET MVC: API

- Method to override:
  - CanReadType
  - CanWriteType
  - WriteToStreamAsync
  - ReadFromStreamAsync

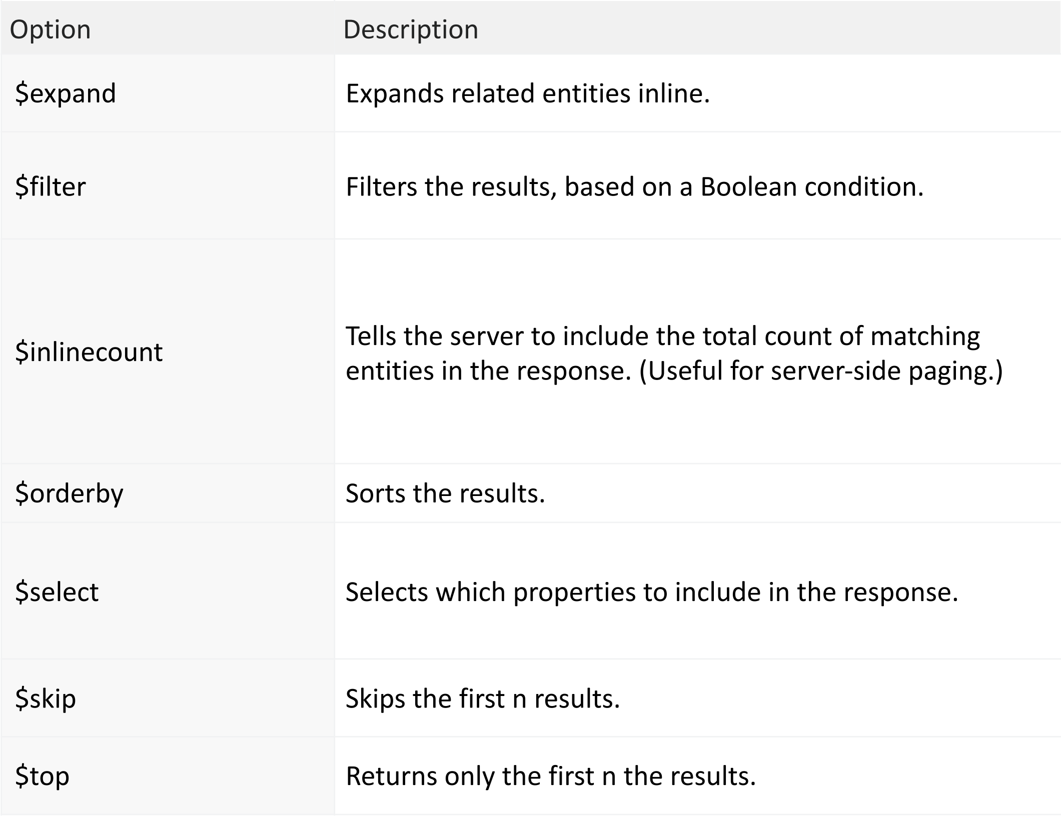
- Register mediaType
  - Constructor: SupportedMediaTypes.Add(new MediaTypeHeaderValue("text/custom"));
  - Global.asax: config.Formatters.Add(CustomFormatter)

ASP.NET MVC: API

- CSV Media Formatter (only write)

ASP.NET MVC: API

- Odata
  - EnableQueryAttribute



Option	Description
$expand	Expands related entities inline.
$filter	Filters the results, based on a Boolean condition.
$inlinecount	Tells the server to include the total count of matching entities in the response. (Useful for server-side paging.)
$orderby	Sorts the results.
$select	Selects which properties to include in the response.
$skip	Skips the first n results.
$top	Returns only the first n the results.

ASP.NET MVC: API

- Help Page
  - Install-Package Microsoft.AspNet.WebApi.HelpPage

ASP.NET MVC: API

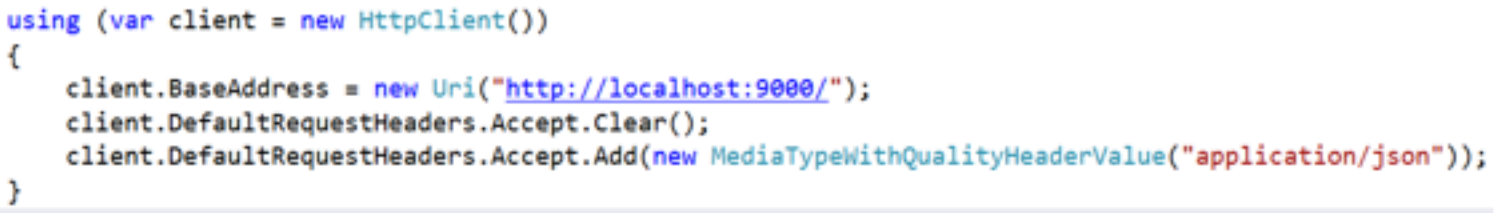
- Dependency Injection
  - Implement IDependencyResolver
  - Register dependencyResolver in global.asax
    - config.DependencyResolver = new MyResolver();

ASP.NET MVC: API

- Refactoring with a repository

ASP.NET MVC: API

- HttpClient
  - Set ContentType header
  - Async / await
  - GetAsync / PostAsJsonAsync / PutAsJsonAsync / DeleteAsync



```
using (var client = new HttpClient())
{
    client.BaseAddress = new Uri("http://localhost:9000/");
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));
}
```